# Truncation attacks on MACs

Chris J. Mitchell
Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK.
c.mitchell@rhul.ac.uk

28th June 2003

## Abstract

A new type of attack on Message Authentication Codes (MACs) is introduced which takes advantage of possible weaknesses in interfaces to hardware security modules. In particular, if a module does not fix the degree of MAC truncation employed, then potentially serious attacks are possible.

## 1 Introduction

MACs, i.e. *Message Authentication Codes*, are a widely used method for protecting the integrity and guaranteeing the origin of transmitted messages and stored files. The sender and recipient of a message (or creator and verifier of a stored file) must share a secret key $K$, chosen from some (large) keyspace. The message to be protected, $D$ say, is input to a MAC function, along with $K$, and the output is the MAC. The MAC is then sent or stored with $D$.

One particularly important class of MAC schemes are the CBC-MACs, so called because the MAC function is based on the 'CBC mode' of block cipher encryption (see, for example, [3]). Suppose the underlying block cipher has $n$-bit blocks and uses a key of $k$ bits. If $X$ is an $n$-bit block then we write $e_K(X)$ (or $d_K(X)$) for the block cipher encryption (or decryption) of $X$ using key $K$. A CBC-MAC on a message $D$ using a block cipher key $K$ is then computed as follows. $D$ is padded and split into a sequence of $q$ $n$-bit blocks: $D_1, D_2, \ldots, D_q$. The MAC computation is as follows.

$$\begin{aligned} H_1 &= e_K(D_1), \\ H_i &= e_K(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q), \text{ and} \\ \text{MAC} &= g(H_q)). \end{aligned}$$

The function $g$ may be a fixed function or it may depend on a key, which may be the same as $K$ or completely independent of it. Note that the MAC used will be truncated to the left-most $m$ bits of the MAC value given in the above equation, where $m \leq n$.

One important type of CBC-MAC is the ANSI retail MAC [1], otherwise known as CBC-MAC-Y or ISO/IEC 9797-1 algorithm 3 [2]. In this scheme $g(H_q) = e_K(d_{K'}(H_q))$ where $K'$ is a second block cipher key. Finally note that, whilst all the discussion here is of CBC-MAC schemes, many of the conclusions apply to other iterative MAC schemes.

## 2 Security considerations

When assessing the complexity of cryptanalytic attacks on MACs, it is normal to quantify the resources needed for the attack in terms of the required numbers of known data string/MAC pairs, chosen data string/MAC pairs, and on-line MAC verifications. Note that MAC verifications are distinguished because, in some environments, it may be easier for an attacker to obtain MAC verifications (i.e. to submit a data string/MAC pair and learn whether the MAC is valid) than to obtain the MAC for a chosen message. In particular, in some cases the legitimate users of a MAC scheme change the key every time a MAC is computed — the dynamic MAC key is itself typically encrypted under a long term key, and stored or sent with the MAC — in such a case, it may still be possible for an attacker to repeatedly submit the same key to the MAC verification process.

Note that there are two main classes of attack on a MAC scheme, namely *key recovery* attacks, in which an attacker is able to discover the secret key used to compute the MACs, and *forgery attacks* in which an attacker is able to determine the correct MAC for a message (without the legitimate key holder having generated it). Key recovery attacks are clearly more powerful than forgery attacks since, once the key is known, arbitrary forgeries are possible. We consider both types of attack in this paper.

## 3 MACs, APIs and usage control tags

In any system using cryptography it is generally accepted as good practice to enforce 'key separation', i.e. to ensure that any particular cryptographic key is used for one purpose only. It is also generally accepted practice to bind a key to one particular cryptographic algorithm, lest a malicious party attempts to learn the value of a key by observing the effects of use of the key with the 'incorrect' algorithm.

This practice is instantiated in the design of cryptographic Application Pro-

gram Interfaces (APIs) for most, if not all, present day hardware security modules (HSMs). Keys stored within such HSMs are typically stored with tags indicating their intended scope of use and the algorithm with which they are to be used. When keys are stored externally to such HSMs they are typically encrypted and integrity protected using 'master keys' held internally to the HSM, and the 'usage control' tags are cryptographically bound to the keys.

In the case of MACs, and in particular CBC-MACs, we consider the scope of such usage control tags. The following parameters would appear to be undeniably within the scope of such control tags:

- the type of data (or keys) over which MACs may be computed using this key,
- the identity of the block cipher to be used to compute the CBC-MAC, and
- the details of the particular CBC-MAC scheme in use.

The inclusion of the first parameter type is a necessary part of standard key management practice — key separation is just as important for MAC keys as it is for encryption keys. Including the second type of parameter is again standard practice — it is extremely dangerous to allow a key to be used with two different cryptographic algorithms. Similar logic applies to the third type of parameter.

The purpose of this paper is to point out that an additional parameter should also be securely bound to a CBC-MAC key. This parameter, namely the degree to which a MAC is truncated, is not always treated in the same way as those above. However, in the remainder of this paper we describe attacks which could be launched against certain types of CBC-MAC scheme if the degree of truncation is not securely bound to a secret MAC key.

## 4 An attack model

The attacks described in this paper all correspond to the case where an attacker is able to request MAC verifications from the holder of the MAC key (e.g. via an HSM API). That is, the attacker is able to submit messages and accompanying MACs and determine whether or not the MAC is correct. As discussed above, in some circumstances such an attack may be much easier to perform than obtaining MACs for chosen messages. For example, such an attack could be launched by injecting trial messages and MACs into a cryptographically protected network, and observing whether these messages are accepted or rejected.

Observe at this point that, using repeated MAC verifications with a fixed message (and working through all possible MAC values), the expected number of MAC verifications required to learn an $m$-bit MAC is

$$2^{m-1} + (2^{m-1} - 1)/2^m$$

i.e. approximately $2^{m-1}$ for $m > 2$. This rather simple observation is the basis of the attacks we describe in the remainder of this paper.

From this point on we suppose that the attacker is also able to specify the MAC length; that is the attacker is able to determine the degree of truncation of the MAC. Such a situation is less likely to be realistic for a communications network (where the MAC length is likely to be fixed) but could be feasible if the attacker has temporary access to the API of an HSM, and the API is not designed to fix the degree of truncation for a key.

Specifically we suppose that the attacker is able to specify the degree of truncation for any multiple of some integer $r$ up to $n$; for simplicity we also suppose $r|n$. In a typical case we might have $r = 8$ and $n = 64$, and the attacker is then able to obtain a MAC verified for any length $m$ from the set $\{8, 16, 24, 32, 40, 48, 56, 64\}$.

## 5 A simple MAC forgery attack

It is simple to see (and very well known) that short MACs are subject to a trivial forgery attack based on repeated MAC verifications. This is based on the observation made in section 4 that a correct $m$-bit MAC can be learnt using approximately $2^{m-1}$ MAC verifications. Hence it is standard practice to choose $m$ large enough so that such threats are not realistic in practice. However, if the attack model described in section 4 applies, then such a precaution may not be effective.

To see this we describe a forgery attack which works against any MAC mechanism, no matter how it is computed. To simplify the attack description, suppose that the legitimate users are using full length MACs, i.e. $m = n$. The attacker, having chosen a message for which a forged MAC is required, first uses an expected $2^{r-1}$ MAC verifications to learn the first $r$ bits of the correct MAC for this message. Next, the attacker submits for verification a series of $2r$-bit guessed MACs (all starting with the correct first $r$ bits) until the correct string is found. The expected number of MAC verifications will again be simply $2^{r-1}$. Repeating this process it is straightforward to see that the expected number of MAC verifications required to learn the complete $n$-bit MAC will be just $(n/r)2^{r-1}$. In the example given above, i.e. where $r = 8$ and $n = 64$, this means that a 64-bit MAC could be forged using just $2^{10} = 1024$ MAC verifications.

## 6 An enhanced key recovery attack on the ANSI retail MAC

The second attack we consider is based on that described in a recent paper [4]. This attack operates against the ANSI retail MAC. Specifically, [4] describes a key recovery attack requiring $(\lceil n/m \rceil + 1)2^{(n+m)/2-1}$ MAC verifications. (Note that this attack is based on an idea due to Preneel and van Oorschot [5], except that MAC verifications are used instead of known MACs.)

The point we make here is that, in the attack model described in section 4, this attack can be launched even if the legitimate users use a large value of $m$. Specifically, regardless of the actual value of $m$ in use, in this attack model a key recovery attack will require $(n/r + 1)2^{(n+r)/2-1}$ MAC verifications. In the case where $r = 8$ and $n = 64$ this would mean that 'only' $2^{38}$ MAC verifications would be required to launch a key recovery attack. In fact this key recovery attack will work just as effectively against any CBC-MAC scheme fitting the definition given above.

## 7 General observations

We conclude this paper by making two general observations.

Firstly, any attack on a MAC scheme (key recovery or forgery attack) that requires $2^t$ known $n$-bit MACs can, in the above attack model, be converted to an attack requiring $(n/r).2^{t+r-1}$ MAC verifications. This follows immediately from Section 5. E.g., if $r = 8$ and $n = 64$, then $2^{t+10}$ MAC verifications would be required.

Secondly, any MAC scheme attack (key recovery or forgery attack) that requires $2^{n/2}$ known $n$-bit MACs to find a pair of 'colliding' values of $H_q$ can be converted to an attack requiring just $(n/r + 1)2^{(n+r-2)/2}$ MAC verifications. This is based on observations made in a [4], and the argument goes as follows.

The attacker first generates a sequence of messages by some means, and for each generated message $M$ uses a MAC verification to test whether or not $\mathrm{MAC}(M) = 0^r$, where $0^r$ denotes a block of $r$ zero bits. If this equation holds then the message $M$ is stored, and such a message should be found on average once every $2^r$ messages tried. Using an argument precisely analogous to that in section 5, the attacker now uses an (expected) further $(n/r - 1)2^{r-1}$ MAC verifications to learn the complete $n$-bit MAC for $M$. The total number of MAC verifications required to learn the $n$-bit MAC for $M$ is thus simply $(n/r + 1)2^{r-1}$. It is important to observe that the $n$-bit MAC for $M$ will have its first $r$ bits set to zero.

The above process is repeated until a total of $2^{(n-r)/2}$ messages have been found for which the complete $n$-bit MAC is known and for which the first $r$ bits of each MAC are all zeros. The usual 'birthday probability' arguments (see, for example, [3]) say that there is a better than even chance that two of the messages will have identical MACs, and hence will 'collide' in their values of $H_q$.

Finally, observe that finding such a set requires a total of $(n/r+1)2^{(n+r-2)/2}$ MAC verifications, as claimed above. That is, if $n = 64$ and $r = 8$, the attack requires a total of $2^{38}$ MAC verifications instead of $2^{32}$ known MACs. This type of argument is behind the result in section 6.

# References


[1] American Bankers Association, Washington, DC. *ANSI X9.19, Financial institution retail message authentication*, August 1986.

[2] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher*, 1999.

[3] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.

[4] C. J. Mitchell. Key recovery attack on ANSI retail MAC. *Electronics Letters*, 39:361–362, 2003.

[5] B. Preneel and P.C. van Oorschot. A key recovery attack on the ANSI X9.19 retail MAC. *Electronics Letters*, **32**:1568–1569, 1996.